AI Artificial Intelligence in Design '98

Fifth International Conference on
Artificial Intelligence in Design

Instituto Superior Técnico, Lisboa, Portugal

Pre-Conference Workshops: 18–19 July 1998
Conference: 20–23 July 1998

workshop 5 notes

emergence in design

Convenors:
Scott Chase University of Sydney, Australia
Linda Schmidt University of Maryland, USA

Fifth International Conference on Artificial Intelligence in Design (AID'98) is sponsored by:
Key Centre of Design Computing, University of Sydney, Australia
Instituto Superior Técnico, Lisboa, Portugal

Emergence in Design

A workshop to be held in conjunction with
Artificial Intelligence in Design'98
19 July 1998, Lisbon, Portugal

"We don't know what we're looking for until we find it, so we have to find a way to look for it!"

An identifiable design property which has not been explicitly represented can be said to be emergent. Discovering and using the emergent properties in design is regarded as emergence in design. Recognizing emergent phenomena is an important part of innovative or creative design and like innovation or creativity, it resists definition. Much has been written about emergence in the context of cognitive science and visual representation, but little exploration has been done with regard to design in general, with the bulk of the research in shape grammars. This workshop aims to bring together design researchers to discuss topics relating to emergence in design, including

- identifying examples of emergence;
- describing systems which support emergence and the representations necessary to achieve emergence;
- methods to identify emergent phenomena; and
- the advantages and disadvantages of emergence.

Workshop Topics

Discussion will address a number of questions; among those are:

- What do we mean by emergence?
- How does design emergence differ from design evolution?
- What different types of emergence can one identify? For example, is it possible to describe functional emergence?
- Is a system that supports emergence necessarily preferable to one that doesn't? What about the ambiguity that emergence can introduce?
- Are emergent-savvy systems a realistic goal for design researchers?

The workshop will be structured around succinct keynote presentations (selected from the submissions) followed by open discussion. Our discussion will be directed to articulate an answer to one or more of the key questions posed above.

Position Papers

We seek position papers or extended abstracts addressing any of the above questions, or describing design systems which are capable of supporting emergence. Papers which offer examples of emergence in design are strongly encouraged. Papers should be no more than seven pages in length. Submissions will be reviewed by the international advisory committee, and will be selected on the basis of their contribution to the topics under discussion.

Convenors

Scott Chase, University of Sydney
Linda Schmidt, University of Maryland

Advisory Committee

Adam Borkowski, Polish Academy of Sciences
Mark Gross, University of Colorado, Boulder
Jeff Heisserman, The Boeing Company
Terry Knight, Massachusetts Institute of Technology
Kumiyo Nakakoji, Nara Institute of Science and Technology

Participants (as of 8 July 1998)

Can Baykan, Middle East Technical University	cbaykan@vitruvius.arch.metu.edu.tr
David Brown, Worcester Polytechnic Institute	dcb@cs.wpi.edu
Esteban de la Canal, Universidad Nacional de La Plata	steve@sol.info.unlp.edu.ar
Scott Chase, University of Sydney	scott@arch.usyd.edu.au
Enxi Chou, University of Wales Aberystwyth	eec@aber.ac.uk
John Gero, University of Sydney	john@arch.usyd.edu.au
Jeff Heisserman, The Boeing Company	jeff.heisserman@boeing.com
Peter Matthews, Cambridge University	pm131@eng.cam.ac.uk
Jeanette McFadzean, The Open University	j.mcfadzean@open.ac.uk
Rivka Oxman, Technion	arrro01@tx.technion.ac.il
Linda Schmidt, University of Maryland	lschmidt@eng.umd.edu
Kristina Shea, EPF-Lausanne	kristina_shea@epfl.ch

Tentative format and agenda

Sunday, 19 July 1998
8:45–12:45

To assure smooth running of the workshop, we would like all workshop participants to prepare for participation in three ways:

- Read the papers submitted to the workshop. These can be found at `http://www.arch.usyd.edu.au/~scott/AID98/emergence-workshop/papers`.
- Prepare an exceedingly brief set of comments that can be titled: "What Emergence Means to Me: Definition and an Example".
- Review the discussion topics listed in the Workshop Announcement and determine which one you would most like to see addressed during our time together.

A tentative agenda for the workshop follows.

8:45–9:00	Self Introduction	All Attendees
9:00–9:15	Workshop Welcome and Format	Chase and Schmidt
9:15–10:25	2 to 5 minute Round of Presentations of *"What Emergence Means to Me: Definition and an Example"*	All Attendees
10:25–10:45	"What Emergence Means to Me" by Proxy	Chase and Schmidt
10:45–11:00	Break	All Attendees
11:00–11:15	Identification of Key Discussion Question(s)	Chase
11:15–12:15	Group(Small Group) Discussion of Key Question(s)	All Attendees
12:15–12:35	Report Back to Full Group on Discussion(s)	Group Reporters
12:35–12:45	The Last Word	Chase and Schmidt

Functional Emergence
A Position Paper

David C. Brown
AI in Design Research Group,
Computer Science Dept.,
WPI, Worcester, MA 01609, USA

http://www.wpi.edu/~dcb/
dcb@cs.wpi.edu

> *An identifiable design property which has not been explicitly represented can be said to be emergent. Discovering and using the emergent properties in design is regarded as emergence in design.* [Chase & Schmidt 1998]

First, let me suggest a rewording of the above implied definition that I think is an important refinement:

> An identifiable design property which has not been explicitly **anticipated or explicitly represented in the current (partial) design** can be said to be emergent.

If a property was "anticipated" it wouldn't have that 'surprise' quality that's characteristic of emergence. That is, if you knew it was coming, it isn't a surprise, even if it isn't explicitly represented.

However, I realize that this a weak distinction. In the classic example of four squares being placed together to produce an extra square that they enclose, that extra square still appears, even if you know it's coming. Perhaps we might call that **Expected Emergence**?

If a property has "not been explicitly represented" anywhere in the available knowledge of the person or system then it cannot be directly "identifiable". Putting three squares together to enclose a triangle, for example, isn't significant if you aren't able to recognize it as a triangle. That is, you need to already have knowledge of triangles. If you can match the emergent design property directly with existing knowledge, let's call this **Directly Identifiable** emergence.

It is possible to imagine a discovery process that identifies a particular property of a design as "interesting"—to use the AM term [Lenat 1982]—and therefore worth remembering, classifying and naming. For example, with no prior knowledge of triangles you might be able to discover that emergent shape and label it. Whether you could use it once found depends on whether what you discover is classifiable, and whether the properties of it's type can be inherited. Let's call this **Indirectly Identifiable** emergence.

Clearly, identification can be done *deliberately*, perhaps driven by a goal that encourages active openess during the design process—such as might occur when someone is trying hard to be creative, as opposed to just 'going through the motions'. In

this situation one can imagine both directly and indirectly identifiable emergence: people would take the time to use discovery processes.

Identification can also be done in a more passive/unconcious way, where associations between stored knowledge and the ongoing design trigger matching, leading to directly identifiable emergence. Such 'noticing' requires some 'priming', either by explicitly making a mental note [Adelson & Soloway 1985] or by having recently used knowledge available due to some other task or subtask.

The matching present in identification is also part of the Analogical Reasoning process, an important aspect of creativity in design [Goel 1997]. However, it's argued that for analogy the matching requires abstraction in order to be possible. For example, to make an analogy between pipes and wires you need to have appropriate abstractions of both water and electricity so that they match sufficiently.

It's hard to imagine this being done as automatically as a "see a square, know it's a square" match. Hence, analogical reasoning might be one of the discovery processes that provide indirectly identifiable emergence.

In addition to analogical reasoning, functional reasoning [Umeda & Tomiyama 1997] (i.e., more conceptual level reasoning) is seen as a key component of creativity—although you can be "routine" at any level [Brown 1996].

Functional reasoning in design is concerned with the *intended use* of the design, its purpose. While the *analogical use* is interesting (e.g., a shoe used as a hammer, or a pen used to make a hole in a piece of paper) it isn't designed in. It's usually discovered, due to a need, by analogical reasoning (and possibly by experimentation). One might consider this to be a kind of **emergent functionality**, as it is revealed after the design has been instantiated.

In [Balazs & Brown 1998] we use analogy to reason about function. For our work we consider function to be the **use of a design's properties by the environment**.

The "environment" might be a person, or some other system. The word "use" is intended to imply that there is some purpose for the interaction, and "property" is defined as loosely as it has been above. For example, a property might be:

- a 'state change', such as a movement,
- 'shape', such a sharp edge,
- 'color',
- 'rigidity', or
- 'weight'.

Under what circumstances would a function be emergent during the design process? Probably…

> An identifiable function of a design which has not been explicitly anticipated or explicitly represented in the current (partial) design can be said to be emergent.

Of the examples above, it's not hard to imagine indirectly identifiable emergence of function due to shape, color, rigidity or weight. And 'shape' is clearly capable of providing directly identifiable emergence of function, in an analogous fashion to the four squares example given above.

However, the extra issue here is to establish "use", or more precisely the capability of use. One obvious possibility is that these are noticed during mental simulations of the design that are intended to check whether the desired properties and behaviors of the design have been successfully incorporated.

Any additional functionality noticed can lead to the retrieval and use of new knowledge, in a fashion similar to analogy. For example, if one notices that the designed object when used is likely to leave some sort of mark, and that's not a bad thing, then one might consider incorporating some of the structural or behavioral characteristics of pens.

Consequently, having an emergent function is not only possible, but may even be beneficial, especially if creativity is being sought.

References

B. Adelson & E. Soloway, The Role of Domain Experience in Software Design, *Trans. on Software Engineering*, IEEE, Vol. SE-11, No. 11, 1985, pp. 1351-1360.

M. E. Balazs & D. C. Brown, A Preliminary Investigation of Design Simplification by Analogy, *Proc. Artificial Intelligence in Design '98*, Lisbon, Portugal, July 1998.

D. C. Brown, Routineness Revisited. *Mechanical Design: Theory and Methodology*, (Eds.) M. Waldron & K. Waldron, Springer-Verlag, 1996, pp. 195-208.

S. Chase & L. Schmidt, Call for Papers, Emergence in Design workshop, AID98, *http://www.arch.usyd.edu.au/~scott/AID98/emergence-workshop/* Lisbon, 1998.

A. Goel, Design, Analogy, and Creativity, *IEEE Expert*, Vol. 12, No. 3, May/June 1997.

D. B. Lenat, AM: Discovery in Mathematics as Heuristic Search. *Knowledge-Based Systems in Artificial Intelligence*, (Eds) R. Davis & D. B. Lenat, McGraw-Hill, 1982, pp. 3-225.

Y. Umeda & T. Tomiyama, Functional Reasoning in Design, *IEEE Expert*, Vol. 12, No. 2, March/April 1997.

Emergence in Designing

John S Gero
Key Centre of Design Computing
Department of Architectural and Design Science
University of Sydney
NSW 2006 Australia
john@arch.usyd.edu.au

Introduction

It has long been recognised by the Gestalt psychologists that emergence is a phenomenon exhibited by humans. What has not been well known or well understood is what its is in designing and what role it may play. This contribution aims to lay the groundwork for a discussion of these topics. It commences with a brief introduction to design as exploration, which is followed by an outline of some recent insights into designing, which relate to emergence, insights derived largely from cognitive studies of designers. This is followed by a brief introduction to two foundational concepts for emergence: situatedness and constructive memory.

Designing as Exploration

Designing as exploration takes the view that the state space of possible designs to be searched is not necessarily available at the outset of the design process. Here designing involves finding the behaviours, the possible structures and/or the means of achieving them, ie. these are only poorly known at the outset of designing (Logan and Smithers 1993). Exploration may be viewed in two ways. It may be viewed as a form of meta-search: the designer searches for state spaces amongst the set of possible predefined state spaces. It may be viewed as a form of construction where each new state space bears some connection to the previously constructed state space(s). This form of exploration cannot be reduced to meta-search. Exploration connects with the ideas of conceptual or non-routine designing: not specifying or even being able to specify at the outset all that needs to be known to finish designing. Designing has been recognized as belonging to the class of problems called "wicked" problems (Rittel and Webber 1973) which have these characteristics.

Recent Insights into Designing

Recent studies of the cognitive behaviour of designers has indicated that other processes occur in designing which are not included in these traditional models of designing. Of particular interest here are two concepts: "reflection in action" and "emergence". The first of these refers to the notion that a designer does not simply design and move on but rather reflects on what he or she has done and as a consequence has the capacity to reinterpret it. Schon (1983) has called this a designer "carrying out a conversation with the materials". Emergence, which is a directly related concept to reflection, is "seeing" what was not intentionally put there (Holland 1998, Gero 1996). Implicit in these important ideas are the seeds for what will be described later.

Protocol analysis is the primary tool for examining such cognitive processes in designing (Eckersley 1988, Gero and McNeill 1998, Goldschmidt 1991, Schon and Wiggins 1992, Suwa and Tversky 1996, Suwa, Gero and Purcell 1998). Schon and Wiggins (1992) found that designers use their sketches as more than just external memory, they used them as a basis for reinterpretation of what had been drawn: this

maps on to emergence and theirs and other studies provide strong evidence for this form of designing. Suwa, Purcell and Gero (1998) have found that designers when sketching revisit their sketches after a while sometimes make unexpected – emergent – discoveries, Figure 1. They concluded that "sketches serve as a physical setting in which design thoughts are constructed on the fly in a situated way".

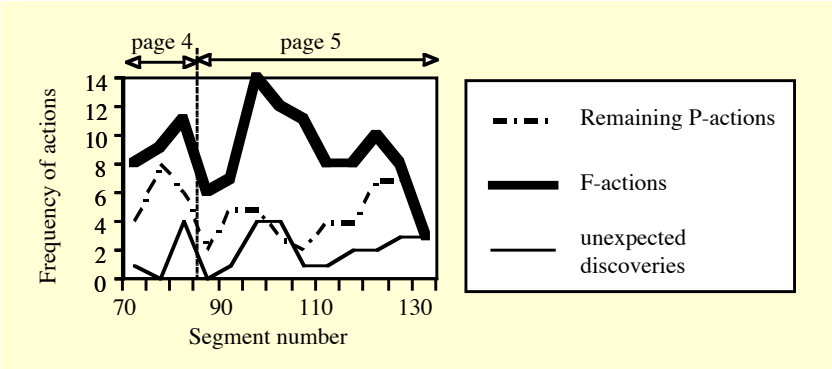


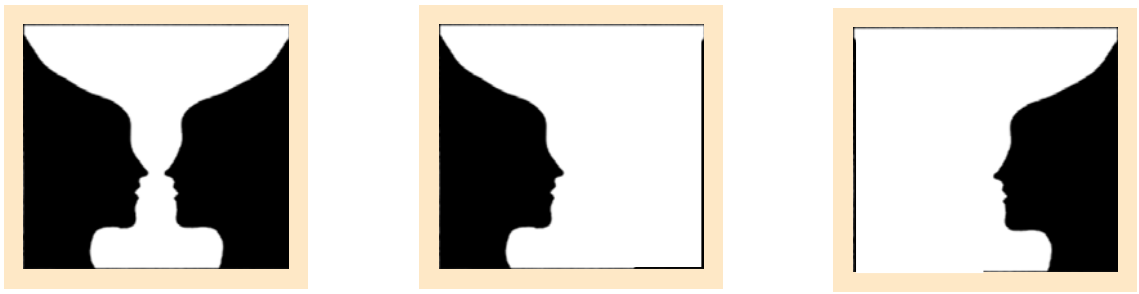
Figure 1. Correlation between unexpected discoveries and functional cognitive actions (F-actions) as opposed to purely perceptual actions (P-actions) in a design session. Segment number refers to the segments in the protocol and the page number refers to the pages of sketching (Suwa, Purcell and Gero 1998).

Reflection and emergence have increasing evidentiary support from protocol studies of designers generally (Suwa et al 1998).

Situatedness and Constructive Memory

The lack of the current models to describe adequately our current view of designing has brought the need to develop models which include such concepts as reflection and emergence and processes which match those of exploration. Work in cognitive science and related areas has developed two sets of ideas that have the capacity to augment, rather than displace, our current models to bring them closer to our needs. The two sets of ideas fall under the areas of "situatedness" and "constructive memory".

Situatedness (Clancey 1997) holds that "where you are when you do what you do matters". This is in contradistinction to many views of knowledge as being unrelated to either its locus or application. Much of artificial intelligence had been based on a static world whereas design has as its major concern the changing of the world within which it operates. Thus, situatedness is concerned with locating everything in a context so that the decisions that are taken are a function of both the situation and the way the situation in constructed or interpreted. The concept of situatedness can be traced back to the work of Bartlett (1932) and even earlier to that of Dewey (1896) who laid the foundations but whose ideas were eclipsed for a time. Figure 2 demonstrates the concept of situatedness through an example of emergence. The designer has to be in the situation where both heads appear at the same time for the white vase to emerge, Figure 2(a), otherwise the designer may never emerge a vase and may never use that emergent figure in later designing. Figure 2(b) shows only one head at some time. The other head may also appear at another time when the first head is not there, Figure 2(c), but unless the situation exists when they both appear at the same time no emergent vase appears.



(a) (b) (c)

Figure 2. (a) Two dark human-like heads in profile, reflections of each other create the situation where a white vase can be seen to emerge; (b) a single dark human-like head on the same background does not create the same situation and therefore no emergent vase can be found; (c)) even the other single dark human-like head on the same background facing the other way does not create the same situation and therefore no emergent vase can be found.

Constructive memory (Rosenfield 1988) holds that memory is not a static imprint of a sensory experience that is available for later recall through appropriate indexing. Rather the sensory experience is stored and the memory of it is constructed in response to any demand on that experience. In this manner it becomes possible to answer queries about an experience which could not have been conceived of when that experience occurred. "Sequences of acts are composed such that subsequent experiences categorize and hence give meaning to what was experienced before" John Dewey (1896). This view of memory fits well with the concept of situatedness. Thus, the "memory" of an experience may be a function of the situation in which the question, which provokes the construction of that memory, is asked. This accords well with the experimental results obtained by observing the behaviour of designers (Cross, Christiaans and Dorst 1996). Constructive memory fits well with our notion of emergence and its role in designing. Figure 3 shows a graphical model of constructive memory driven by emergence.

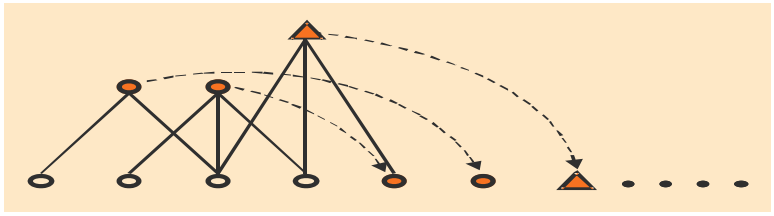


Figure 3. The original design representation (experience), ○, is used to produce emergent features of the design, ●, then the original and emergent features are added as new representations and may be used later in the emergence of new features, ▲, and so on.

These two short introductions to situatedness and constructive memory with its implicit situatedness, suffice to allow us to now utilise these ideas in the development of our understanding of designing to the point where we can begin to include emergence in a model of designing.

Emergence in a Model of Designing

There are three possible classes of emergence in designing when utilising the Function–Behaviour–Structure model. These classes are: emergent structures; emergent behaviours; and emergent functions. Emergent structures are the most commonly held examples of emergence, particularly visual or graphical emergence, such as in the

example in Figure 2. However, behaviour emergence is very common, particularly in architecture. Derived from graphical images behaviours represented as patterns are commonly emerged. These patterns have such labels as symmetry, axiality, and linearity. Higher level patterns can also be emerged. These higher level patterns have such labels as movement, balance and rhythm. Emergence of function is not well understood at all although it is possible to give examples of it.

This presentation will demonstrate the role of emergence in designing through a model of constructive memory and situatedness. It will provide examples of structure and behaviour emergence.

References

Bartlett, F. C. (1932 reprinted in 1977) *Remembering: A Study in Experimental and Social Psychology.* Cambridge University Press, Cambridge

Clancey, W. J. (1997) *Situated Cognition.* Cambridge University Press, Cambridge

Cross, N., Christiaans, H. and Dorst, K. (eds) (1996) *Analysing Design Activity,* Wiley, Chichester

Dewey, J. (1896 reprinted in 1981) The reflex arc concept in psychology. *Psychological Review* **3**: 357–370

Eckersley, M. (1988) The form of design processes: a protocol analysis study, *Design Studies* **9**(2): 86–94

Gero, J. S. (1996) Creativity, emergence and evolution in design: concepts and framework. *Knowledge-Based Systems* **9**(7): 435–448

Gero, J. S. and McNeill, T. (1998) An approach to the analysis of design protocols, *Design Studies* **19**(1): 21-61

Goldschmidt, G. (1991) The dialectics of sketching, *Creativity Research Journal* **4**(2): 123–143

Holland, J. (1998) *Emergence.* Addison-Wesley, Reading, MA

Logan, B. and Smithers, T. (1993) Creativity and design as exploration, *in* J. S. Gero and M. L. Maher (eds), *Modeling Creativity and Knowledge-Based Creative Design.* Lawrence Erlbaum, Hillsdale, NJ, pp. 139–175

Rittel, H. and Webber, M. (1973) Dilemma in a general theory of planning. *Policy Sciences* **4:** 155–160

Rosenfield, I. (1988) *The Invention of Memory.* Basic Books, New York

Schon, D. (1983) *The Reflective Practitioner.* Harper Collins, New York

Schon, D. and Wiggins, G. (1992) Kinds of seeing and their functions in designing, *Design Studies* **13**(2): 135–156

Suwa, M., Gero, J. S. and Purcell, T. (1998). Analysis of cognitive processes of a designer as the foundation for support tools, *in* J. S. Gero and F. Sudweeks (eds), *Artificial Intelligence in Design'98,* Kluwer, Dordrecht, pp. 229-248

Suwa, M., Purcell, T. and Gero, J. S. (1998) Macroscopic analysis of design processes based on a scheme for coding designers' cognitive actions. *Design Studies* **19** (to appear)

Suwa, M. and Tversky, B. (1996) What architects see in their design sketches: implications for design tools, *Human Factors in Computing Systems: CHI'96,* ACM, New York, pp. 191–192

Using a Guideline Database to Support Design Emergence: A Proposed System based on a Designer's Workbench

Peter C. Matthews
Engineering Design Centre
Cambridge University Engineering Department
pm131@eng.cam.ac.uk

Abstract

This paper proposes a system that can suggest possible areas of a design where due to the interaction of various objects, emergent properties might be observed. This system will be based on the design requirements, along with the context they arise in, being passed as a search probe to a guideline database. The designer then browses through the results of the database search for relevant guidelines—some of which might suggest properties resulting from interaction between various design objects, which might be emergent design properties. By identifying these properties (which could be either beneficial or adverse) early on in the design process, any effects these properties might have can be dealt with before prototype testing, thus reducing design time and cost.

1 Introduction

This paper proposes an 'emergence-savvy' designer's workbench. This will be achieved by using a requirements capture tool that interfaces with a guidelines database. This database forms a large knowledge-base covering aspects of design that a designer might not be immediately aware of. When the designer adds a requirement to the design, this will be passed to the guidelines database as a search query. The results of this guidelines search are then presented to the designer for further consideration in the design process. It remains the designer's choice which of these guidelines to use.

This system suggests areas where emergent behaviour might occur due to the interaction of various design objects that the designer might not have considered. This will be illustrated using an example of novice designers building an autonomous guided vehicle.

2 Evolution v Emergence

For the purpose of this paper, *evolution* will be defined as the expected refinement of the design as it progresses. Design *emergence* will be defined as the unexpected aspects that arise due to the interaction of different objects. These definitions will be illustrated using the Cambridge University Engineering Department's second year undergraduate Integrated Design Project.[1]

Teams of six students (typically subdivided as 2 mechanical specialists, 2 electronic and 2 software) are required to design, build and test a semi-autonomous vehicle (see Figure 1) that is able to navigate a course (marked out by a white line) and perform various object handling tasks [Clarkson et al., 1998]. Each vehicle is constructed from a fixed

[1] A more complete description of the IDP is given in the AID'98 conference proceedings [Ball et al., 1998]

kit of parts (with no restrictions on bulk materials such as steel or wood) and must be completed in four weeks.

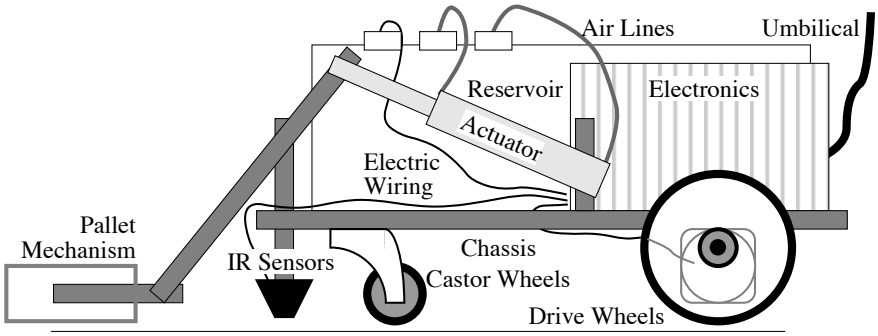


Figure 1: An example of an IDP vehicle

In the design of this vehicle, both design evolution and emergence occur. Using the infra-red sensors as an example of evolution, the design of the electronic interface between the sensors and the controlling computer has evolved from the need of the controller to be able to read the sensors. A simplistic example of emergence occurs in the layout and positioning of this sensor array. Often, teams will place the sensor array near the drive axle of the vehicle. The vehicle navigates using three systems: the mechanical (sensor positioning and drive wheel placement), electronics (sensor reading), and software (control routines). Although each system may behave as expected, the interaction of these results in a feedback delay which result can in poor navigation (in this case, it is mainly dependent on the distance between the sensor array and the driving wheel axle). As this is an unexpected behaviour arising from object interaction, the distance between sensor array and drive axle becomes a new design property, explicitly represented as a guideline.

3 Overview of the System Components

The proposed emergence support system contains three components: a designer's workbench, a requirements capture tool, and a guideline database. The designer's workbench is the computer interface used to capture the design and the guideline database includes a search engine for guideline retrieval. The requirements capture tool provides the link between the guideline database and the design capture tool (see Figure 4).

3.1 Designer's Workbench

Blessing [Blessing, 1994] has proposed PROSUS, a framework for capturing design process data as part of the design information via matrix of events at each node of the product tree (Figure 2). Each matrix is formed from lists of general design issues and generic design activities. An individual design event (or series of events) can then be pigeon-holed according to its position in the product tree, the type of issue and the way it is being addressed. This method forms the basis of the designer's workbench. The design nodes are structured according to the Engineering Design Centre's in-house

developed Common Product Data Model (CPDM).

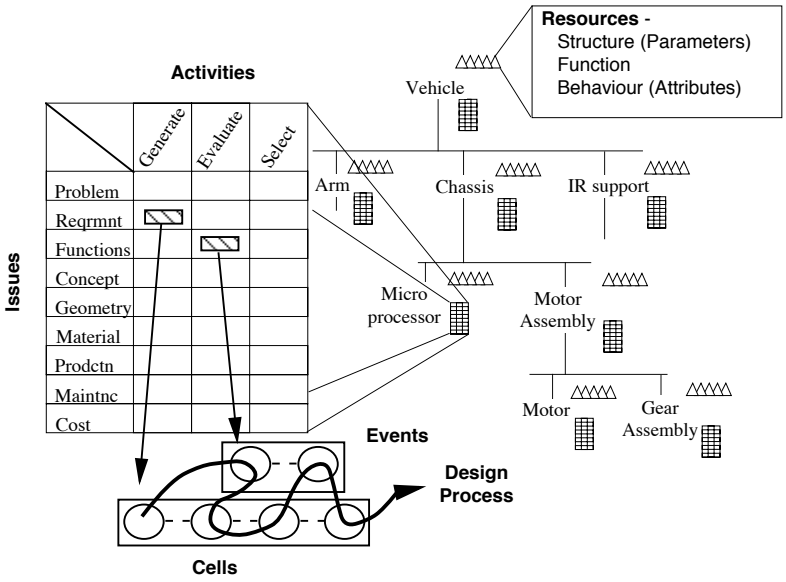


Figure 2: The PROSUS model as used within the CPDM

The traditional mechanism for grouping product data is provided by the Bill of Materials. This divides the product into assemblies, sub-assemblies, and individual parts. The CPDM is an object-oriented version of this structure with a number of added properties. In the C++ implementation of the CPDM, the base object class (the *artefact*) contains property and characteristics lists. These lists encapsulate design information related to the artefact such as weight, dimensions, or constraints within the design object (the information on these lists is dynamically extendible). The artefact class is then cast to the design hierarchy classes of product, assembly, part, and component (Figure 3). There is also an interface class that permits data propagation between artefacts. The artefact class also contains a list of alternatives for that node. These are other possible design solutions (though not necessarily complete) for that particular node.

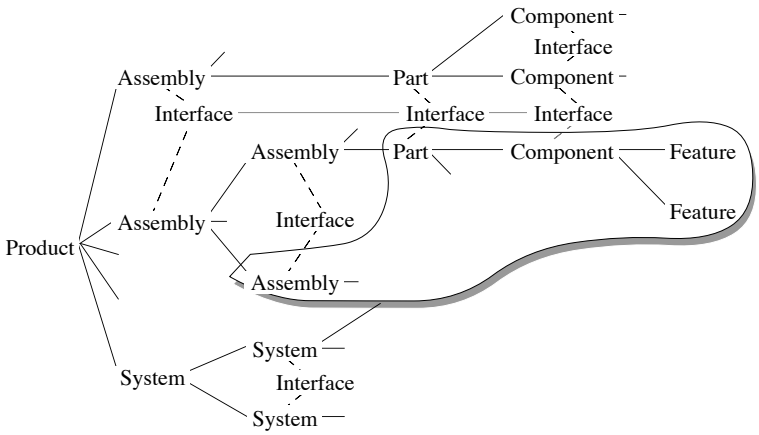


Figure 3: The CPDM class hierarchy

3.2 Requirements Capture

The requirements, or specifications, of a design are the guide to what the designer must achieve. These requirements should not be a static set laid down at the start of a design, but must evolve as the design progresses. Hollins and Pugh [Hollins and Pugh, 1990, Pugh, 1990] give a general overview as to what these requirements should include and suggest 35 areas to be considered when generating requirements.

There are a number of requirement capture tools commercially available, e.g. SLATE [TD Technologies, 1998] and DOORS [QSS, 1998]. Cambridge University Engineering Department have implemented a Specification Capture tool (*SpecBuilder*) based on the suggested method of Pahl and Beitz [Pahl and Beitz, 1996]. This tool is only used at the start of a design project, and has little scope for evolving. A CPDM class for specification data has been proposed permitting the capture of further requirements during the design.

3.3 The Guidelines Database

A guideline is a context-sensitive prescriptive recommendation for some action given some issue of concern [Nowack, 1997]. Guidelines can provide additional assistance by predicting likely outcomes of actions and by identifying additional issues that should be considered. For guideline support to be effective, appropriate guidelines must be available to the designer at the time of a design decision. Nowack's approach was developed into a tool useful for small companies with a need to access a wide range of knowledge that could be contained in a collection of guidelines.

A catalog of 3500 such guidelines has been compiled in [Edwards et al., 1993]. Using this set of guidelines with a protocol study of the design of a wall-mounted optical device, it was demonstrated that guidelines can mirror the design process [Charlton et al., 1997]. The study showed that in the case of an incomplete guideline database, protocol analysis is a good method of adding new guidelines to the database.

4 Proposed System

The proposed system would be based around the implementation of Blessing's designer's workbench, PROSUS [Blessing, 1994, Ball et al., 1998]. As the design is refined, further requirements are added to the problem, either by requirements breakdown, or as new issues arise from the design process. These requirements will be entered from the workbench by invoking a requirements capture tool (e.g. *SpecBuilder* as mentioned in Section 3.2). The requirements that are generated will be stored within the design artefact where they were generated.

These new requirements would be passed to the guideline database (see Figure 4), along with some context gathered from the design artefact's parent (weighted such that the higher up in the hierarchical structure the less effect it will have on the search probe). The database then carries out a text-based retrieval for relevant guidelines for that requirement. The designer would then be able to browse relevant guidelines and select a

subset of these which would apply to the design artefact. Some of these guidelines will possibly bring up unexpected design properties, i.e. emergent properties.

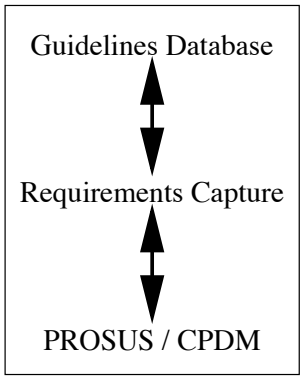


Figure 4: The interactions of the system components

This system supports emergence by assisting the designer in discovering aspects of, or interactions within, the design which might otherwise have been overlooked, or only discovered at a later time. Consider the AGV example as described in Section 2. The design initially has a top level requirement of 'navigate along the painted white lines'. Later in the design, when the designer has decided to use infra-red sensors arranged in an array for navigation, the further requirement is added to the design: 'sensor array to be attached to chassis'. When the guideline database receives this information along within the context of the higher level requirement, it returns the guideline: 'navigation sensors must be placed as far ahead as possible for the most stable results'. This property was not at first known in the design (at least to novice designers), and hence can be considered emergent according to the definition used in this paper.

5 Conclusions

This paper has presented a system for suggesting possible emergent properties. These properties can be either beneficial or adverse to the design. The system uses requirements and their context within the design to propose possible properties that the designer would not necessarily have thought of. This provides an 'emergence-suggesting' system that draws the designers attention to aspects that might not otherwise be immediately clear in the early design stages. Such a system would save design time and cost by reducing the amount of prototype feedback to the design.

References

[Ball et al., 1998] Ball, N. R., Matthews, P. C., and Wallace, K. M. (1998). Managing conceptual design objects: An alternative to geometry. In Gero, J. S. and Sudweeks, F., editors, *Artificial Intelligence in Design '98*. Kluwer, Dordrecht.

[Blessing, 1994] Blessing, L. T. M. (1994*). A Process-Based Approach to Computer-Supported Engineering Design*. PhD thesis, University of Twente.

[Charlton et al., 1997] Charlton, C. T., Nowack, M. L., and Wallace, K. M. (1997). Engineering design guideline support scheme. In Riitahuhta, A., editor, *Proceedings of the 11th International Conference on Engineering Design*, volume 2, pages 657–660. Tampere University of Technology.

[Clarkson et al., 1998] Clarkson, P. J., Isgrove, D. W., Matheson, J. M. R., Parks, G. T., and Wilmshurst, T. (1998). 1B Integrated Design Project: Mobile robot. Cambridge University Engineering Department.

[Edwards et al., 1993] Edwards, K. L., Wallace, K. M., and Aguirre-Esponda, G. (1993). Designers' electronic guidebook — an engineering design guidelines database. Technical Report CUED/C-EDC/TR14, Cambridge University Engineering Department.

[Hollins and Pugh, 1990] Hollins, B. and Pugh, S. (1990). *Successful Product Design*. Butterworths.

[Nowack, 1997] Nowack, M. L. (1997*). Design Guideline Support for Manufacturability*. PhD thesis, Cambridge University Engineering Department.

[Pahl and Beitz, 1996] Pahl, G. and Beitz, W. (1996*). Engineering Design: A Systematic Approach*. Springer-Verlag London, second edition.

[Pugh, 1990] Pugh, S. (1990). *Total Design*. Addison-Wesley.

[QSS, 1998] QSS (1998). Quality Systems and Software — company web site. URL: `http://www.qssinc.com/`.

[TD Technologies, 1998] TD Technologies (1998). Company web site. URL: `http://www.tdtech.com/`.

Schema Emergence in Design

Rivka Oxman
Faculty of Architecture and Town Planning
Technion- Israel Institute of Technology
Haifa, Israel 32000
e-mail: arrro01@tx.technion.ac.il

In the following notes, I briefly review our current research in schema emergence and experimental work in the development of a design aid system which is intended to support the emergence of new schema through the interactive manipulation of graphical representations. In this extended abstract I illustrate our approach as a means of addressing certain of the general questions related to the subject of emergence.

1. Introduction: shape emergence vs design emergence

are there different types of emergence?

Restructuring and reinterpretation of images is a fundamental property of design thinking. This reasoning is generally facilitated by the interaction between the designer and the visual representations of the design. The process of recognizing new emergent properties within existing representations characterizes emergence.

Most current research in design computation deals with graphical emergence in *shape* interpretation. Stiny's pioneering work, (Stiny, 1980, 1993) formalized emergence through shape grammars. A general framework for the description and the representation of the emergence of shapes has recently been proposed by (Soufi and Edmonds, 1995). They developed a formulation of categories of emergent shapes and a mechanism for the isolation of those shapes based upon a proposed computational framework. Another significant development is work based on the recognition of implied shapes within line drawings (Liu, 1995). Interpretations of shapes and the interpretation of patterns of shapes into graphical structures are a further advance in the area of shape interpretations and their semantics (Gero, Damski and Jun, 1995).
However, the human designer knows how to manipulate shapes, because he reasons with certain kinds of knowledge structures. The *Electronic Cocktail Napkin* (Gross 1996, Gross and Do, 1996) is a prototype constructed on top of a freehand drawing program that explored how computer-based sketching programs can provide an enhanced environment for form recognition as well as for access of case knowledge in the architectural domain.

In our work we also deal with the problems of shape representation, recognition and reinterpretation. However, rather than dealing with specific classes of shape emergence (e.g. geometric classes) we are attempting to understand and model how the emergence of cognitive knowledge structures of higher level semantics can be supported by the syntax of shapes within generic representations.
We propose a cognitive approach through which generic knowledge, a well formalized body of specific knowledge, can contribute to the emergence of new generic schema by shape manipulation and re-interpretation.

2. Schema emergence as a paradox of generic design

how does design emergence differ from design evolution?

Developing and manipulating generic knowledge is one of the most significant forms of cognitive behavior of the designer. Generic design demands knowledge handling properties related to the schema and variables which are manipulated in generic design (Oxman and Oxman, 1991). The emergence of new schema is a fundamental cognitive capability of creativity in the human designer. A paradox of creative design is how the human designer can discover new schema while working with the generic content of existing schema. Dickemann, (Dickemann, 1930) illustrated how a transformation process can occur in which specific prototypes of chairs can be transformed to other profiles. This and other works raise an interesting question: *how can specific typological knowledge contribute to the emergence of new types in creative thinking?* Schema emergence thus appears to be a unique, and highly significant, form of emergence in the research literature. Our research attempts to cognitively model this class of emergence.

Within the context of generic processes we employ the term typology as design domain knowledge of classes of design problem types. One of the most significant schematic representations which designers employ in visual development of design representations is knowledge of the type (Oxman and Oxman, 1991). Typologies can exploit generic representations which are specific to the typological class. Thus each typology implies the existence of a schema of generic representations. Typological knowledge, therefor, is characterized by a set of generic representations which are associated with specific design problem types, and the knowledge of the variables of the type is organized in a hierarchical order of representations of which the highest level is that of the schematically represented class description. Generic design is the exploitation of this structured knowledge in design reasoning.

Typologies are well known in the context of evolutionary design. However, exploration process in which new types emerge, the employment of generic design is not yet well understood. In our modeling of this process, we propose that the designer can decide how he wishes to reformulate, or re-structure, the graphical representation, and thus exploit existing schema and generic processes in creative design (Oxman, 1998). For example, in the case of chair design, the typology of the chair can be represented as a holistic component or by various combinations of sub-components. Secondly, within each particular element of structure, parametric modification is also possible as a means to differentiate the image. For example, the following figures illustrate two analyses of possible representations which are derived from the same typologies. In figure 1 the typology has resulted in the ladderback chair by C.R.Mackintosh. In figure 2, the same typology has resulted in a bench-based chair by Maichele de Luchi.

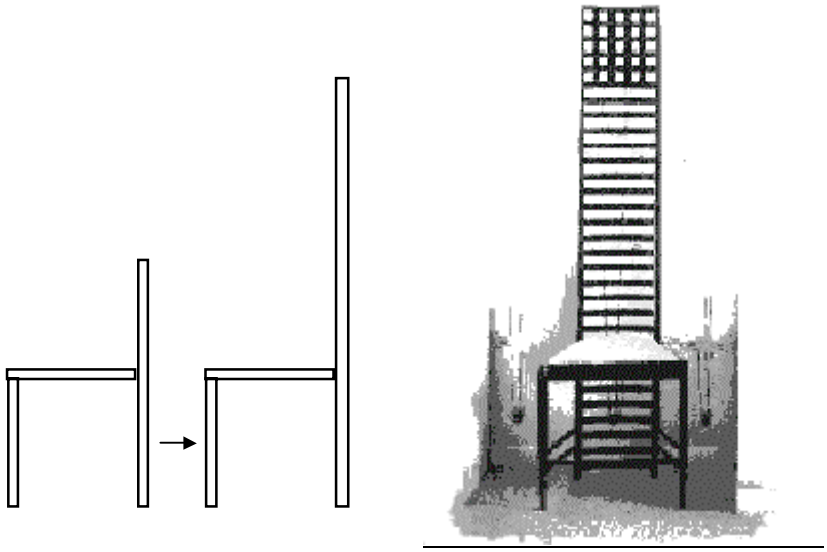


Figure 1. The 'Ladderback chair by C.R.Mackintosh' associated with generic representation of 'tall proportion' (analysis done by Zvi Zyit and Aviram Kuri)

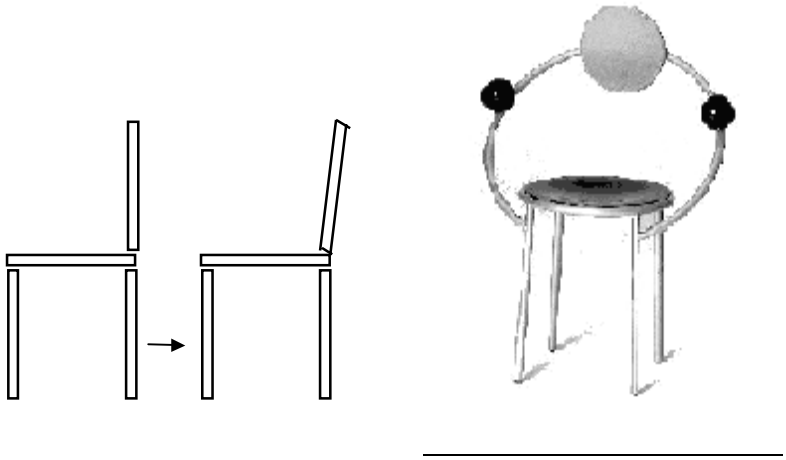


Figure 2. The 'First chair by Maichele de Lucchi' associated with generic representation of a 'benched-based chair' (analysis done by Zvi Zayit and Aviram Kuri)

Generic knowledge enables transformations to take place when creating such sub-types through parametric and formal manipulations within the generic schema of the type. Recent works have already dealt with representation and manipulation issues of typological knowledge in generic design in the architectural domain (Oxman and Oxman, 1991, Achten and Oxman, 1998). In the present work, we are investigating how the same knowledge which is associated with generic and typological design can contribute to the emergence of new designs. In this paper we investigate the phenomenon of *schema emergence* in the re-representation process. We present a model of the emergence of new schema. According to this view, we view the emergence of schema as a the result of an exploration process of representations in which sub-types of one generic

schema can be re-interpreted as a sub-type of another generic schema. Each schema representation can be re-structured, or componentized, in different ways so that another relevant new schema representation can emerge.

3. Design system which supports schema emergence: a computational approach

what do we mean by emergence in computational terms?

While much research in emergence and creativity in design is either theoretical, or too specific for an application, we are trying to study how to build computational environments supporting human emergence in design.

In our approach, the computational environment behaves as a graphical interactive design medium which is supportive of the cognitive capabilities of the designer. Schema emergence is supported by providing an interactive interface which assists in the construction of new structures which can be derived from existing ones. The representational system operates through the maintenance of generic schema and typological knowledge while enabling modifications within the type. The typological generics act within the background while the designer interacts with the representation dynamically to achieve transformations. Once the limits of a typological schema have been reached by the designer, he is free to transform the typological schema and then explore variations within the new typological framework.

We are implementing a computational system for *schema emergence* which requires a relevant generic schema and typological knowledge suitable to a problem class. We have employed a general terminology which we hope will address our ideas in our current implementation. We utilize the three abstract terms, objects, actions and meta-actions, in order to describe computational emergence of design schema.

- Objects: these are objects in a modeling language which define the internal generic representation of a typological object. .
- Action: is any transformation resulting in a change of parameters and re-compilation of the internal representation.
- Meta action: when the code of a specific modeling language is changed and reconstructs a new or modified structure of the internal representation.

Emergence can be defined as follows: initial schema is a given initial scene which contains basic and accepted code of objects related to a specific type. Transformations and modifications are changes of parameters of a specific type. Emergence can occur when multiple interpretations of a specific structure provide options to change the code. A change of code can occur by importing new code of new types which may change the structural organization and the content of the internal representation.

The current system is implemented in VRML. We are currently developing an interface to support the human emergence. The designer interacts with the typological representation dynamically to achieve transformations. Once the designer can recognize the emergence of another typological schema, he is free to change and modify the internal representation of the new schema and then explore another set of variations within the new typological framework.

The user interface presents to the user the internal representation and allow him to act directly on the internal representation for further explorations and modifications.

Acknowledgments
This work is supported by a DFG grant from the German Government. The general objective of the grant is to study Media-Based Support for Design Creativity. Professor Bernd Streich of the Universities of Kaiserslautern and Bonn is my German counterpart in the research. I am grateful for our discussions which have been stimulating and important for the development of my thinking.
Hezi Golan from the Faculty of Computer Sciences, Technion and Eyal Nir from the Faculty of Architecture and Town Planning, and Daniel Brainin from Industrial Design at the Faculty of Architecture and Town Planning, Technion. are currently involved in this research and have contributed to these ideas and their implementation.

References

Gross, M.: 1996, 'The electronic cocktail napkin - working with diagrams' Design Studies, Vol. 17

Gross M. and Do E. and Zimring C.: 1994, Using diagrams to access a case base of ar−chitectural designs in *AID'94* (ed, J. Gero) Kluwer Academic

Gero S. and Damski J, and Jun H. (1995) Emergence in CAAD systems, in CAAD Futures '95 National University of Singapore, Tan M. and The R. (eds.) pp.423 - 438

Liu T.Y. (1995) Problem decomposition on restructuring shapes in terms of emergent sub-shapes, in CAAD Futures '95 National University of Singapore, Tan M. and The R. (eds.)pp.455-468

Oxman Rivka (1998) Beyond sketching: visual reasoning through re-representation in cognitive design media, in CAADRIA '98 Sasada T., Yamaguchi S. Morozumi M. Kaga A. and Homma R. (eds.) Kumamoto University, pp. 337-347

Oxman R. E. and Oxman R. M, (1991) Refinement and adaptation in design, Design Studies.

Soufi B. and Edmonds E. (1995) A framework for the description and representation of emergent shapes, in CAAD Futures '95 National University of Singapore, Tan M. and The R. (eds.)pp. 411-422

Stiny, G. (1980) Introduction to shape and shape grammars. Environment and Planning B, Vol. 7, pp. 345-351

Stiny G., (1993) Emergence and continuity in shape grammars, in Flemming U. and Van Wyk (eds.) CAAD Futures, '93, Elsevier, Amsterdam, pp. 37-54

Function Driven Form-Making: Revealing New Functionality Through Parsing

Linda C. Schmidt
Department of Mechanical Engineering
University of Maryland
College Park, MD 20742-3035 USA
lschmidt@eng.umd.edu

Scott C. Chase
Department of Architectural and Design Science
University of Sydney
NSW 2006 Australia
scott@arch.usyd.edu.au

1 Introduction

Form-making to the craftsman is function-satisfaction to the engineer. Grammars that capture a style of form (e.g, Palladian villas [Stin78], Hepplewhite chair backs [Knig80] and Diebenkorn paintings [Kirs86]) are the envy of the mechanical engineer whose designs proceed from function [Flem92] and whose forms are slaves to mechanical function. Generative design for engineering requires adequately describing a user's needs, articulating the space of all possible devices that meet the need, articulating each device in terms of function (ability to meet user needs) and form (geometry), and searching through that space to select good options. Researchers have presented grammar-based generative algorithms that design from functional description to form, but their computational expense as knowledge-based systems invites questions as to their ultimate usefulness. A discussion of recent grammar applications in mechanical design research can be found in [Schm96].

Beyond their potential as a means to automate routine design tasks or explore new design realms, generative design systems that record complete design data can act as guardians of proprietary design knowledge. A company's grasp of design knowledge is always tenuous. Design engineers acquire knowledge through experience and preserve their accomplishments using the design tools available to them. Non-geometric data becomes decoupled from the design and can be lost or exits the company with each retiring or departing designer. As novices replace the more experienced designers, they need access to complete design information. New designers must quickly assimilate the artistry of their predecessors and the skills manifested in prior designs. Function-driven generative design systems may provide key to interpreting functional design information so useful to designers.

2 Function-Driven Form-Making: A Grammar-Based Approach

Function-driven design requires an expression of functional intent, the ability to decompose overall function into more fundamental units for the synthesis into new functional representations, and some means of manipulating component functional units into useful expressions. The difficulties with this function-driven, top-down approach are the

access to knowledge necessary to refine designs into physical artifacts and the ability to represent the complex functional relationships like function-sharing and functional interactions [Flem92].

The key to surmounting these difficulties is finding the means to represent functions so that their usefulness in the design can be recognized computationally. If you adapt a linguist's point of view where a particular set of designs is represented by a language, you can view the rules for design synthesis as a grammar. To understand designs represented in your language, you would need an interpretation process, which we call parsing. Parsing here means resolving an expression in a language into its component parts and describing those parts in terms of the grammar rules on which the language is founded.

3 Function Parsing and Examples

Functional parsing algorithms are, in a sense, the reverse of function-based generation algorithms. Function-based design generation algorithms build designs based on functional knowledge. Function parsing algorithms extract functional knowledge from an existing design. This allows the design to be classified by its functional characteristics. Being able to extract information from a design that reflects the function structure upon which it is built is the basic tenet of reverse engineering.

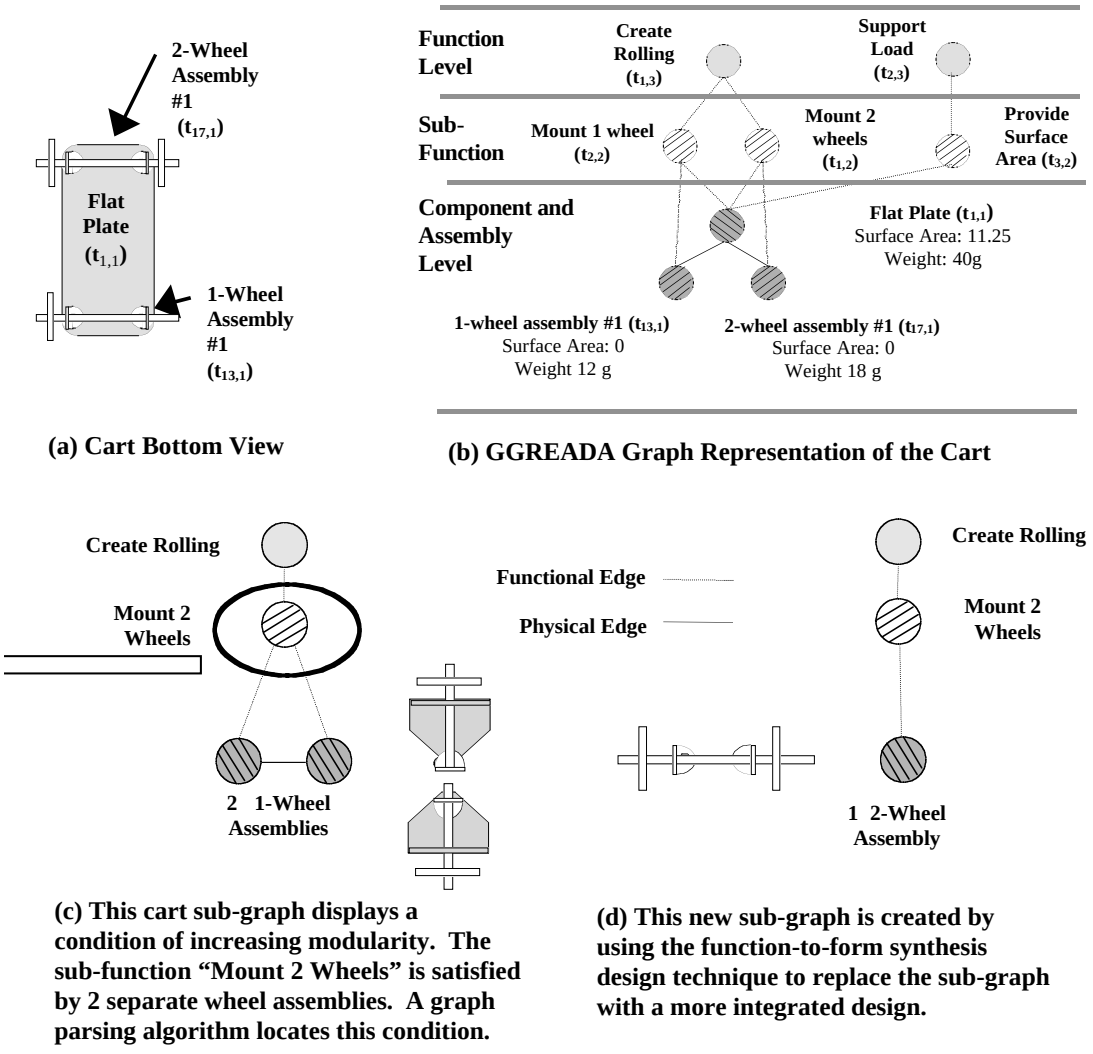
A truly powerful language of designs does not represent each design as combinations of predefined parts. If that were true, all our mechanical designs would resemble Rube Goldberg machines using the same set of basic function-executing mechanical assemblies. Instead, operators can be constructed that can reveal new entities in the design, i.e. those implicitly represented from the explicit description of other design entities (e.g., feature recognition). The best operators at performing this design maneuver lie somewhere in the mind of good designers. They are able to look at a design and see opportunities to capitalize on existing features of the design to perform new or more efficient functions. This property of some design ideas to grow out of seemingly disconnected, obscured, or hidden characteristics of an existing design is called emergence. Unless a design system can demonstrate the ability to identify emergent functions and forms from its design representations, it will be limited to the capabilities of its explicit design representation and we are back to a Rube Goldberg world.

Recognizing emergent functions and potentially useful functional conditions requires the ability to parse a design into its pertinent functional modules or components and recognize the emergent properties even when they are not explicitly represented. By examining specific cases of function driven design and analysis we can obtain some insight into how functional emergence might be structured. Following are a few such examples.

Example 1: Functional analysis
Little, Wood and McAdam's refined functional analysis approach [Litt97] assimilated a list of basic mechanical functions and the flows they act upon to create a product classification scheme for consumer products, product benchmarking and reverse engineering [Litt97]. With further analysis of the product classification hierarchy, the authors were able to identify key functions as future design priorities. At present, their system is too general for reversal into a product generation method, but is another springboard to generative design work.

Example 2: GGREADA

Figure 1 displays the graph-grammar representation of GGREADA (b) as applied to the design of carts from Mecanno® Set pieces [Schm97]. Cart design sub-graphs are displayed in Figure 1 (c) and (d). Sub-graph (c) modularizes (splits) the function "Mount 2 Wheels" into one that is satisfied by two separate assemblies. Using a parsing algorithm to detect the splitting of functional edges at a node, this functional condition can be located and rectified. The ability to recognize the functional condition of interest re-



(a) Cart Bottom View

(b) GGREADA Graph Representation of the Cart

(c) This cart sub-graph displays a condition of increasing modularity. The sub-function "Mount 2 Wheels" is satisfied by 2 separate wheel assemblies. A graph parsing algorithm locates this condition.

(d) This new sub-graph is created by using the function-to-form synthesis design technique to replace the sub-graph with a more integrated design.

Figure 1 Use of a functional parser to find an emergent condition with the generative graph-grammar of GGREADA [Schm97].

quires that the set of conditions of interest be articulated either exhaustively, or in some recoverable form.

Example 3: Emergent functions from fixed structures

In one discussion on emergence, Gero [Gero96] references a physical shape described by Finke [Fink90] as having at least eight different uses-functions, in our parlance—depending on how use chose to see and use the structure. In Figure 2 we highlight two possible uses, one is an umbrella use (our invention) and the second is a sled use

(Finke's). Both are possible because the physical characteristics of the shape allow it to function in multiple ways. In our example, the relevant physical characteristic of the shape is its net shape quality of being a plane oriented parallel to the ground. When a person stands between the shape and the ground, it provides shelter. When a person stands on top of the shape which is in contact with the ground, it may provide transportation, depending on frictional characteristics of both the shape and the ground. So we see again that the recognition of some physical feature and its link to a function is necessary to allow the designer to traverse the tree depicted in Figure 2 to reach different branches signifying different functional uses of the same shape.

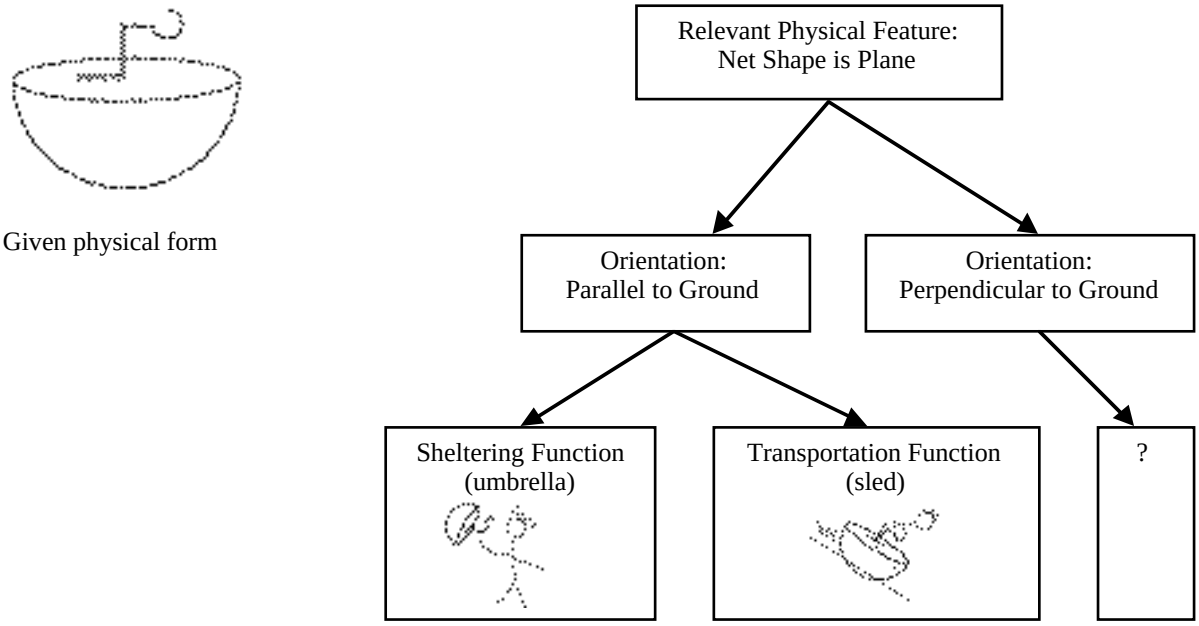


Figure 2 Inventions from emergent functions based on a shape's physical features.

Example 4: Isomorphism grammar detection

In his work on applying grammars to generate mechanism structure graphs, Shetty developed a method for detecting isomorphic planar graphs [Shet98]. He modified a linear asymptotic growth rate algorithm presented by Hopcroft and Wong [Hopc74]. In this algorithm a set of prioritized graph reductions are proposed. Each reduction strictly decreases the size of the graph (as measured by the number of edges and vertices), until it reduces to a regular polyhedron or a single node, each with a one or more labels coding the reductions that led to the end state (Figure 3). These graphs can be tested for isomorphism (as labeled graphs) by exhaustive matching of labels.

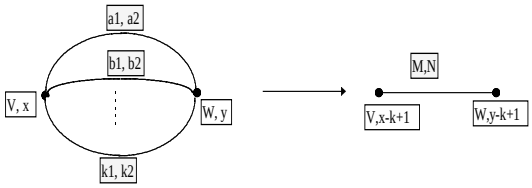


Figure 3 Isomorphism grammar rule for removing clumps – in effect, generalizing the graph into a label edge

This reduction method for isomorphism detection is similar to the abstraction or generalization approach used in Example 3. In that case, by abstracting the shape into its net physical shape, one climbs up a tree of possible function derived from the same feature. One can travel down the tree by a different route to find new functionalities. Keeping in mind that Shetty's mechanism structure graphs are defining forms (which drive function in mechanisms), the isomorphism detection grammar essentially forces the graph into its most abstract state—either a regular polyhedron or a single node—with a set of descriptive labels. If the forms are the same, the resulting abstract graph will be the same. So the abstracting scheme used in the isomorphism grammar is analogous to climbing to the top of the function-form tree in Example 3.

4 Types of Functional Parsing

The four examples of function and form representation indicate two different approaches to parsing for emergence which are here referred to as representation-based and condition-based parsing.

4.1 Representation-Based Parsing

Representation-based parsing relies on articulating the entities of interest in atomic units that are the basis of rules that implement emergence by matching certain combinations of these atomic representation units. The matching that occurs is based solely on the entities matched and not on the context in which they appear. This type of situation is present in the GGREADA example and the functional analysis approach used by Little et al. [Litt97]. In these examples, parsing relies on hard-coding the relationships into the algorithm. In the case of GGREADA, we actually define rolling to be accomplished by mounting at least 3 wheels to the structure. In Little's paper, the devices are defined according to a pre-defined set of functions and sub-functions. Instances of specific functions or sub-functions can easily be retrieved when they are represented explicitly, as in these cases.

4.2 Condition-Based Parsing

Condition-based parsing can also be used in grammar systems. In this method the preconditions for rule application are described as set of logical conditions that must be met as opposed to a description of entities that must be present, i.e., entities that are implicitly described by conditions rather than explicit represented. To parse a design for instances of a condition, an algorithm systematically identifies portions of the design, generalizes the portion, and compares the abstracted result to the required conditions. Here, the conditions provide a context in which the parsing proceeds. Both the isomorphic grammar example and the Finke example seem to use this approach. Condition based parsing can also be found in the GENESIS boundary solid grammar system [Heis94], and in the logic based formulation of conditions which describe spatial relations (called emergent features) described in [Chas97]. In these cases the conditions primarily describe form rather than function, but could also be used for the latter.

5 Conclusion

Here we have abstracted two approaches to parsing designs, toward the goal of discovering emergent functions. As we are in the early stages of this investigation, we do not draw conclusions here, but rather, ask several questions:

- What does each approach require in terms of design representation, techniques, other?
- Is one approach better than the other? What are the advantages and disadvantages of each approach?
- Are there other approaches possible toward the goal of finding emergent functions within a design?

As we progress further in this research, we expect to find the answers to these and other questions.

References

[Chas97] Chase S. C., 1997, "Modeling spatial reasoning systems with shape algebras and formal logic", AI EDAM: Artificial Intelligence for Engineering Design, Analysis and Manufacturing 11:4 1-13.

[Fink90] Finke, R., 1990, Creative Imagery: Discoveries and Inventions in Visualization (Lawrence Erlbaum Associates, Hillsdale, New Jersey).

[Flem92] Flemming, U., J. Adams, C. Carlson, R. Coyne, S. Fenves, S. Finger, R. Ganeshan, J. Garrett, A. Gupta, Y. Reich, D. Siewiorek, R. Sturges, D. Thomas, and R. Woodbury: 1992, "Computational models for form-function synthesis in engineering design," Carnegie Mellon University Engineering Design Research Center Technical Report EDRC, 48-25-92.

[Gero96] Gero, J. S., 1996, "Creativity, emergence and evolution in design", Knowledge-Based Systems, 9:435-448.

[Heis94] Heisserman, J., 1994, "Generative Geometric Design" IEEE Computer Graphics and Applications, 14(2):37-45.

[Hopc74] Hopcroft, J. E. and J. K. Wong, 1974, "Linear Time Algorithm for Isomorphism of Planar Graphs", Proceedings of the 6th Annual ACM Symposium on Theory of Computing, April 30-May 2, 1974, Seattle, Wash., 172-184.

[Kirs86] Kirsch J. L., Kirsch R. A., 1986, "The structure of paintings: formal grammar and design", Environment and Planning B: Planning and Design 13:163-176.

[Knig80] Knight T. W., 1980, "The generation of Hepplewhite-style chair-back designs", Environment and Planning B 7:227-238.

[Litt97] A. D. Little, K. L. Wood, and D. A. McAdams, 1997, "Functional analysis: a fundamental empirical study for reverse engineering, benchmarking and redesign," Proceedings of DETC97, ASME Design Theory and Methodology Conference. Sacramento, CA, September 14-17, DETC97/DTM-3879.

[Schm96] Schmidt, L., and J. Cagan, 1996, "Grammars for Machine Design," J.S. Gero and F. Sudweeks (eds), Artificial Intelligence in Design 96, Kulwer Academic Publishers, 325-344.

[Schm97] Schmidt, L. C. and J. Cagan, 1997, "GGREADA: A Graph Grammar-Based Machine Design Algorithm," Research in Engineering Design, 9:195-213.

[Shet98] Shetty, H., 1998,"A Graph Grammar Approach to the Generation of Non-Isomorphic Graphs for the Structure Synthesis of Mechanism," M.S. Thesis, Mechanical Engineering Department, University of Maryland.

[Stin78] Stiny, G. and W. J. Mitchell: 1978, The Palladian grammar, Environment and Planning B, 5: 5-18